

Элементом html-разметки является служебное слово/символ, которое не только описывает, но и определяет, как будет выводиться информация. В разговорной речи элемент называют **тегом** (тег - это метка), в научной - **дескриптором** (descriptor - «описывающий»).

Синтаксис элементов зависит от типов этих элементов. Существуют:

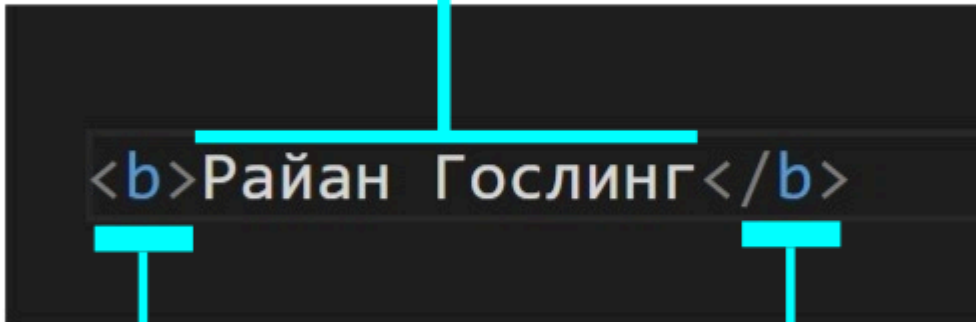
1. Двойные (парные) элементы.

К примеру, элемент `` (bold), который выделяет текст, находящийся внутри него ` жирным шрифтом `. То есть, синтаксис парных элементов состоит из:

- открывающего тега - ``. Синтаксис любого тега состоит из латинских букв (слово или сокращение), располагающегося между знаками "меньше" и "больше" (угловые скобки) ;
- содержания - текста, картинки, ссылки, другого элемента и т.д. ;
- закрывающего тега - ``. Закрывающие теги пишутся со слешем после знака меньше - "<".

Про парные элементы необходимо также знать, что у них всегда есть **содержание**, то есть то, что находится между `` и ``, иначе никаких изменений не произойдёт. Ниже на картинке представлен парный **элемент**:

Содержание



Открывающий тег

Закрывающий тег

2. Одиночные элементы.

Например, элемент `
` ("line break"). Этот элемент переводит нас на новую строку, то есть в текстовом редакторе он как бы аналогичен клавише "Enter". Его особенность в том, что у него *нет содержания* и он самодостаточен.

Но чаще всего одиночные элементы содержат внутри самого тега *атрибуты* или *события*. К примеру, одиночный элемент ``, который помещает на страницу изображение, обязательно должен содержать внутри атрибут `src` ("*source*"), который указывает путь до изображения:

```

```

У `` есть ещё один обязательный атрибут, но о нём будет рассказано позже.

3. Атрибуты

Итак, атрибутом называются действия, составляющие или расширяющие основу элемента. Существуют Уникальные атрибуты, которые свойственны только определённым элементам. Также есть и [Универсальные атрибуты](#), присущие всем элементам, к примеру, атрибут `title` который описывает содержимое при наведении на него курсора (подсказка).

Μια φορά κι έναν καιρό ζούσε
μια πολύ ψηλομύτα ποντικήνα
που κάθε μέρα, πολύ νωρίς,
έβγαινε να σκουπίσει την
είσοδο του σπιτιού της.

Это текст на греческом языке

```
<p title="Это текст на греческом  
языке">Μια φορά κι έναν καιρό ζούσε μια  
πολύ ψηλομύτα ποντικήνα που κάθε μέρα,  
πολύ νωρίς, έβγαινε να σκουπίσει την  
είσοδο του σπιτιού της.</p>
```

Виды элементов

У каждого элемента, кроме служебного действия, есть свои особенности отображения и применения. Кроме того, что существуют парные и

одиноким элементом, они также делятся на **блочные** и **строчные**.

1. Блочные

Базовые особенности блочных элементов:

- а) Начинаются с новой строки;
- б) Занимают всю ширину строки, поэтому...
- в) последующий элемент так же начинается с новой строки.
- г) К ним применимы стилевые (CSS) правила изменения ширины и высоты;
- д) Могут содержать в себе другие блочные и строчные элементы.

Ширина блочного элемента определяется либо вручную, либо шириной элемента, в который он вложен.

2. Строчные

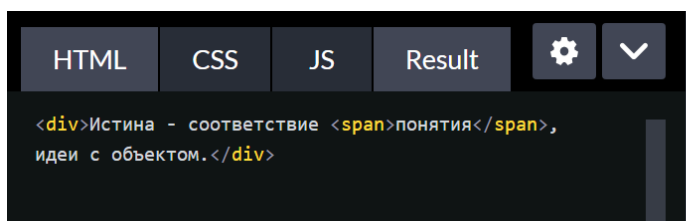
Базовые особенности строчных элементов:

- а) Занимают часть экрана, равную своему содержанию;
- б) Не должны содержать в себе блочные элементы, соответственно:
- в) могут содержать в себе только строчные элементы;
- г) К ним не применимы стилевые правила изменения ширины и высоты.

Для того, чтобы разобраться в этом на примере, возьмём 2 элемента, у которых есть только базовые свойства:

`<div>` - базовый **блочный** элемент;
`` - базовый **строчный** элемент.

Для наглядности зададим элементам фон.



The screenshot shows a web editor interface with tabs for HTML, CSS, JS, and Result. The HTML tab is active, displaying the code: `<div>Истина - соответствие понятия, идеи с объектом.</div>`. The Result tab shows the rendered output: a green rectangular block containing the text "Истина - соответствие" followed by "понятия, идеи с объектом.", where "понятия" is highlighted in pink.

Истина - соответствие **понятия**, идеи с объектом.

Строчный элемент прекрасно уживается внутри блочного.

Если мы сделаем всё наоборот, то получится такой результат:

```
HTML CSS JS Result
<span>Истина - соответствие <div>понятия</div>,
идеи с объектом.</span>
```

Истина - соответствие

ПОНЯТИЯ

, идеи с объектом.

Как видите, предложение было разбито на части внутренним блочным элементом. В данном примере наглядно продемонстрировано и то, что блочный элемент занимает всю ширину строки. По правилам, строчные элементы не могут содержать в себе блочные.

Синтаксис

Табуляция

Любые знаки табуляции в html выводятся как один пробел. Пробелы после открывающего тега и до закрывающего также учитываются, поэтому:

`<p>Солнце хорошо светит</p>` - правильное написание

`<p> Солнце хорошо светит </p>` - неправильное написание

Тренажёр на Степике *чувствителен к табуляции*, поэтому убирайте лишние пробелы.

Вложенность

Вы уже видели, что элементы могут быть вложены друг в друга. В таком случае они становятся родственными (relative). Соответственно, **родительскими** элементами называются те, в которые вложены другие элементы, которые, в свою очередь, называются **дочерними**. Например:

```
HTML CSS JS Result
<div>Родительский элемент
  <div>Дочерний элемент</div>
  Родительский элемент
</div>
```

Родительский элемент

Дочерний элемент

Родительский элемент

Вложенность тегов друг в друга, а также тема **наследования** свойств будут отдельно разбираться в разделе CSS.

Атрибуты и значения

Все атрибуты, прописывающиеся внутри самого элемента, пишутся через **пробел** от самого названия элемента и друг от друга.

Пример:

`<video autoplay loop muted>` - в данном примере у тега `<video>` есть 3 атрибута. У данных атрибутов нет **значений**. Такие атрибуты ещё называются **логическими**.

Если же у атрибута есть значение, то оно записывается после знака равно и помещается в кавычки (двойные или одинарные).

Пример:

`<input type='submit' placeholder='введите имя'>` - в данном примере у тега `<input>` есть 2 атрибута, каждому из которых задано значение. Значение атрибута можно писать без кавычек только если в значении нет пробелов. В данном примере значение атрибута `type` могло быть написано без кавычек - `<input type=submit>`, но это делать не рекомендуется.

У элемента не может быть два одинаковых атрибута, вроде `<input type='text' type='password'>`.

Порядок атрибутов в элементе не имеет значения для отображения.



Следите за кавычками у значений атрибутов.

Одна незакрытая кавычка может вывести из строя весь код.

Базовая структура HTML-документа

Есть некоторые элементы, которые обязательно должны присутствовать в вашем документе. Они составляют каркас страницы.

Выглядит это так:

```
<!DOCTYPE html>                                     language-html
<html>
  <head>
    <meta charset="utf-8">
    <title>Название страницы</title>
  </head>
  <body>
    <!--Содержание страницы-->

  </body>
</html>
```

Отступы при написании тегов в редакторах выставляются автоматически, чтобы была наглядна **вложенность** элементов друг в друга.

`<!DOCTYPE html>` - указывает, что тип документа - HTML5. На старых сайтах можно встретить и другие типы с другим синтаксисом например:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

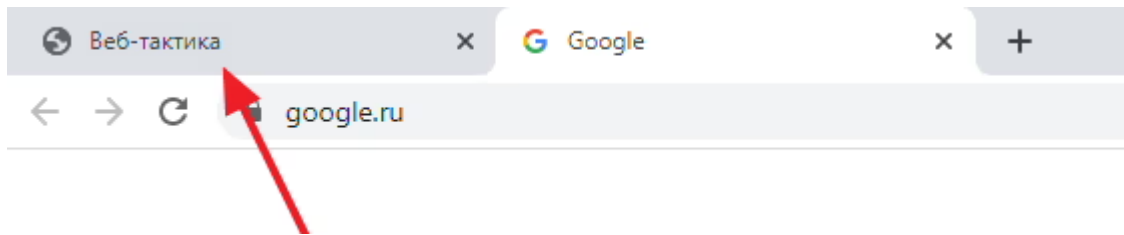
`<html>` - это контейнер, который содержит в себе всю страницу. Браузер прочитает разметку и без этого элемента, но по синтаксическим правилам данный элемент обязателен.

`<head>` - в данном контейнере содержится техническая информация о странице (кодировка, размеры, масштабируемость, стили и т.п.), а также сюда помещается название вкладки данной страницы.

`<meta>` - служебный тег, который редактирует некоторые свойства страницы с помощью атрибутов. К примеру, атрибут `charset` задаёт кодировку страницы. Самая популярная кодировка, поддерживаемая наиболее известными браузерами - это UTF-8 (работает с большинством существующих символов. Если не задать данную кодировку, то некоторые браузеры могут не отображать кириллицу).

В остальном синтаксис тега `<meta>` сводится к парам атрибутов `name` (`property`) - `content`, то есть имя (свойство) - значение

`<title>` - данный элемент содержит название страницы, оно обычно отображается на вкладке браузера, например:



`<body>` - в данном контейнере содержится вся видимая для пользователя на странице информация.



В редакторах базовую структуру можно вставлять по горячим клавишам. Обычно, нужно написать восклицательный знак (!), а затем нажать на клавишу **TAB**. Если в вашем редакторе это не работает, то необходимо установить расширение [EMMET](#) (в CodePen оно уже есть).

`<!-- -->` - тег для комментария. Комментарии не отображаются в браузере и видны только в разметке. Помогают оставлять заметки себе и коллегам. EMMET сокращения для комментария - `Ctrl+/,` в VSCode и некоторых других редакторах - `Alt+Shift+A`.

Например:


```

...<!DOCTYPE html> == $0
<html lang="ru">
  <head>...</head>
  <!-- Отображаемое тело страницы -->
  <body>
    <!-- Оболочка страницы -->
    <div class="wrapper"> flex
      <!-- Шапка с меню гамбургером -->
      <header class="header">...</header>
      <!-- Основной контент -->
      <main class="content">
        <section class="welcome">
          <div class="container">
            <div class="welcome__row"> flex
              <div class="welcome__column">...</div> flex
              <div class="welcome__column">...</div> flex
            </div>
          </div>
        </section>
        <section class="announces">...</section>
      </main>
      <!-- Подвал -->
      <footer class="footer">...</footer>
      <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js"></script>
      <script src="js/script.js"></script>
    </div>
  </body>
</html>

```

Почему так важны пробелы и прочая табуляция?

Объясню на простом примере:

Предположим, что нам нужно сделать переключатель, который отображает, подписались ли мы на контент или нет. Если контент внутри будет равен "Оформить", то при нажатии на кнопку он будет менять на "Отписаться" и наоборот.

Подписка на рассылку

Подписка на рассылку

Оформить

Отписаться

С помощью JavaScript мы будем проверять содержание элемента

(создаёт кнопку) и нужно, чтобы оно чётко совпадало с тем, что мы будем проверять.